

Insertion Sort

For all elements in the list from left to right:

while current element less than left neighbour:

swap elements

Insertion Sort



A horizontal array of six light gray rectangular boxes, each containing a black number. The numbers are 8, 11, 3, 5, 15, and 9, arranged from left to right. The boxes are separated by thin vertical lines.

8	11	3	5	15	9
---	----	---	---	----	---

Insertion Sort



Insertion Sort

Current 8



Insertion Sort

Current 8

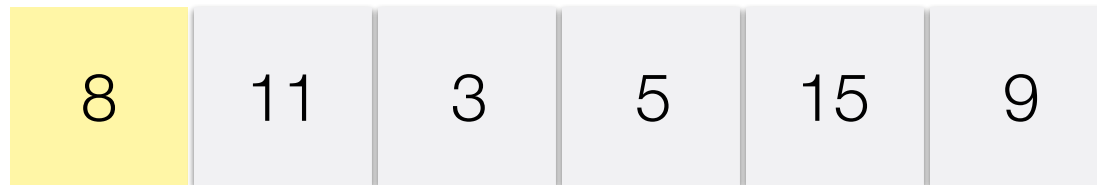


No elements
to the left

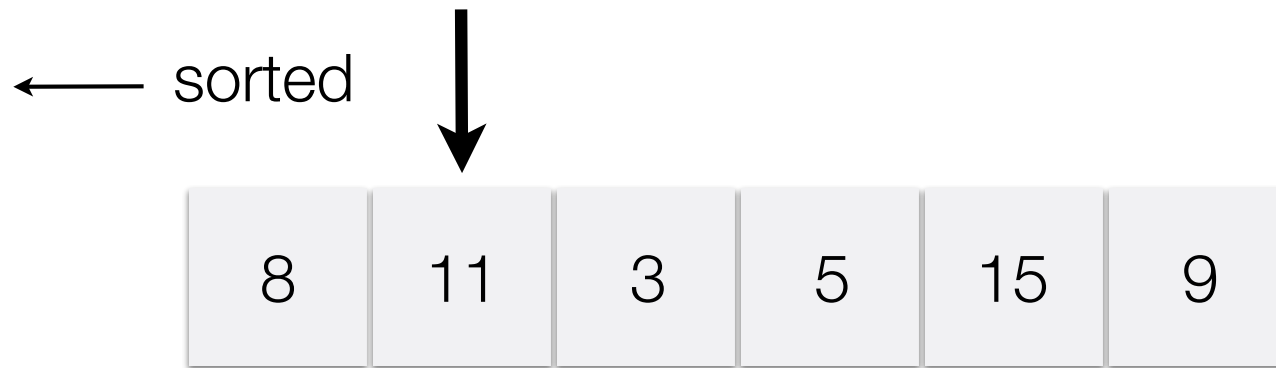
Insertion Sort

Current

8



Insertion Sort



Insertion Sort

Current

11



Insertion Sort

Current

11

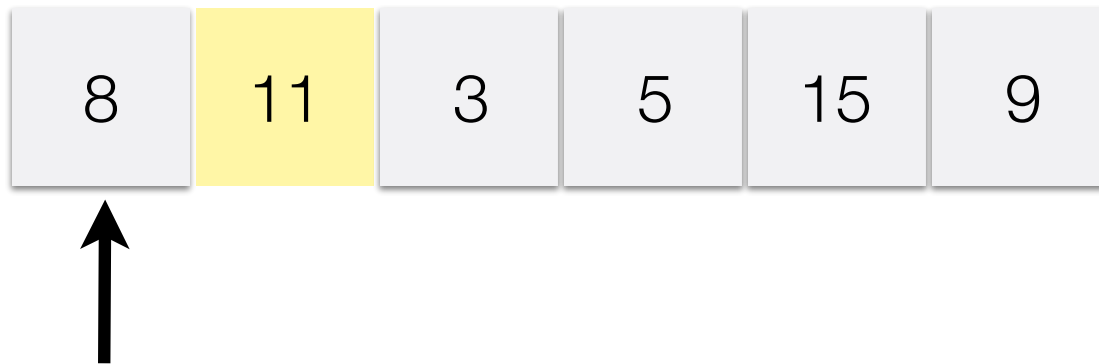


> 11 ?

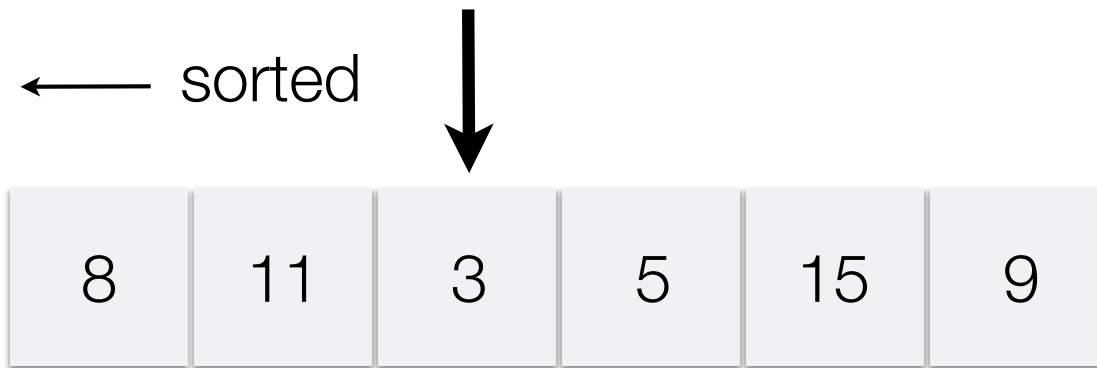
Insertion Sort

Current

11



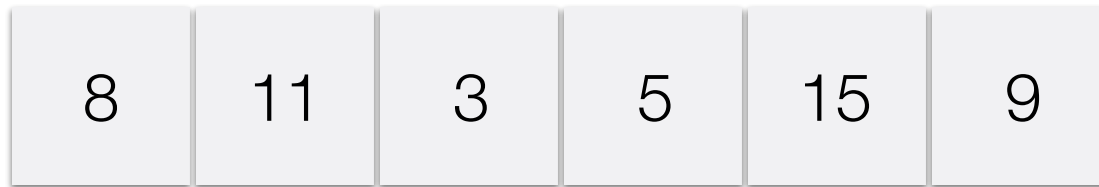
Insertion Sort



Insertion Sort

Current

3



Insertion Sort

Current

3



> 3 ?

Insertion Sort

Current

3



↑
> 3 ?

Insertion Sort

Current

3



> 3 ?

Insertion Sort

Current

3



↑
> 3 ?

Insertion Sort

Current

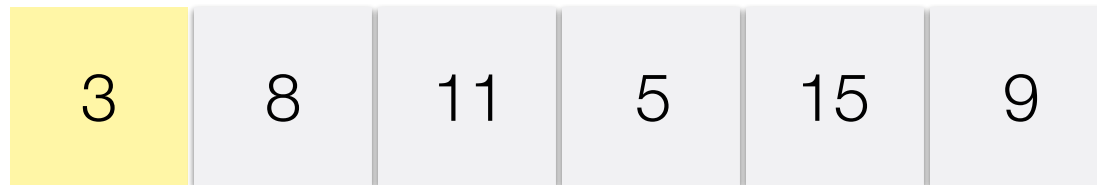
3



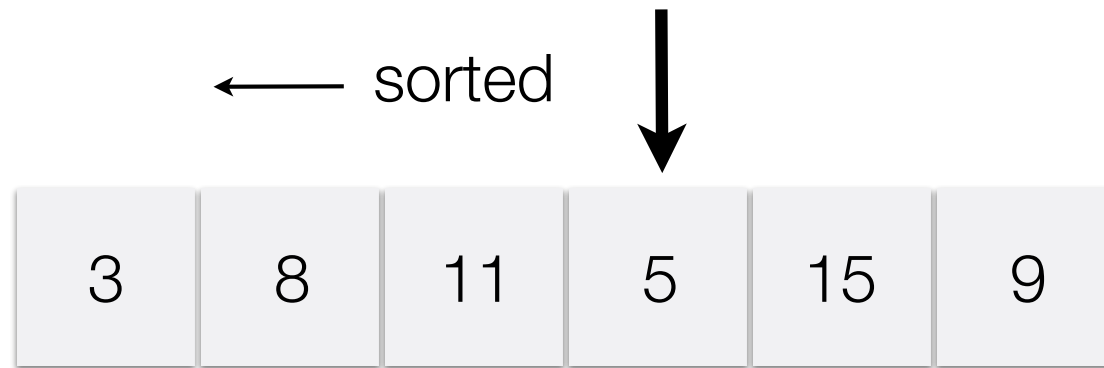
Insertion Sort

Current

3



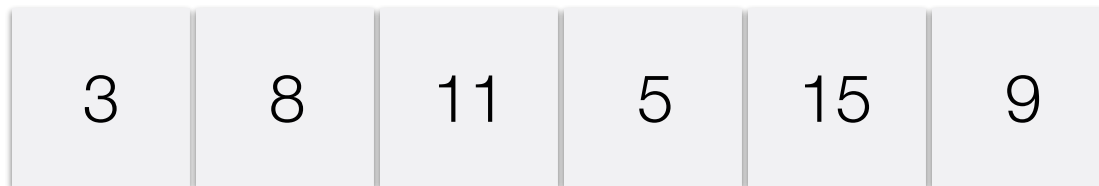
Insertion Sort



Insertion Sort

Current

5



Insertion Sort

Current

5



> 5 ?

Insertion Sort

Current

5



> 5 ?

Insertion Sort

Current

5



> 5 ?

Insertion Sort

Current

5



> 5 ?

Insertion Sort

Current

5

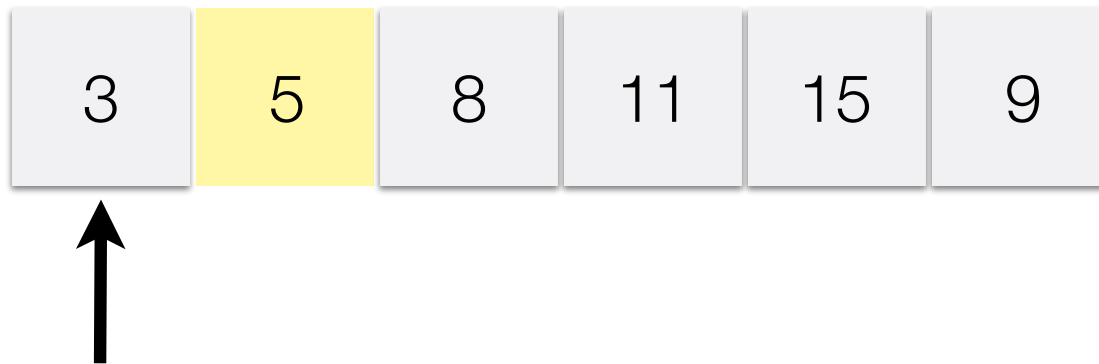


> 5 ?

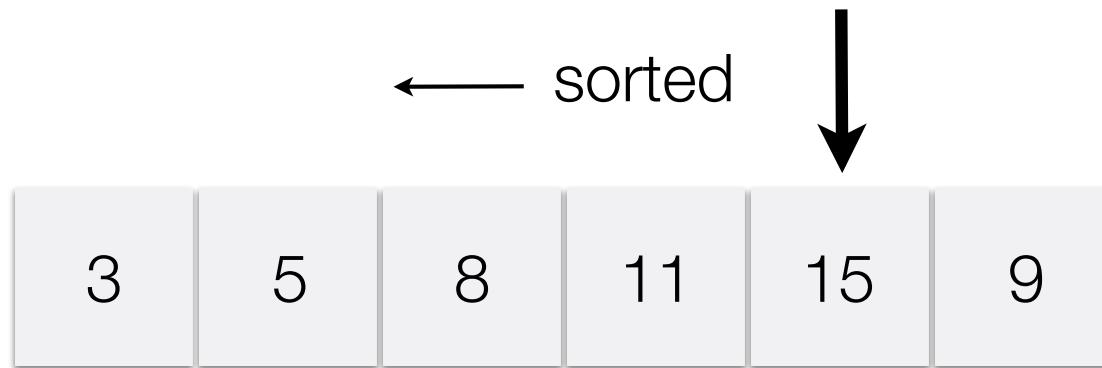
Insertion Sort

Current

5



Insertion Sort



Insertion Sort

Current

15



Insertion Sort

Current

15



> 15 ?

Insertion Sort

Current

15



Insertion Sort



Insertion Sort

Current

9



Insertion Sort

Current

9



> 9 ?

Insertion Sort

Current

9



> 9 ?

Insertion Sort

Current

9

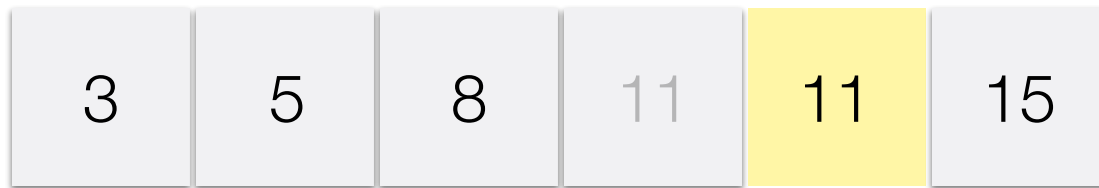


> 9 ?

Insertion Sort

Current

9



> 9 ?

Insertion Sort

Current

9

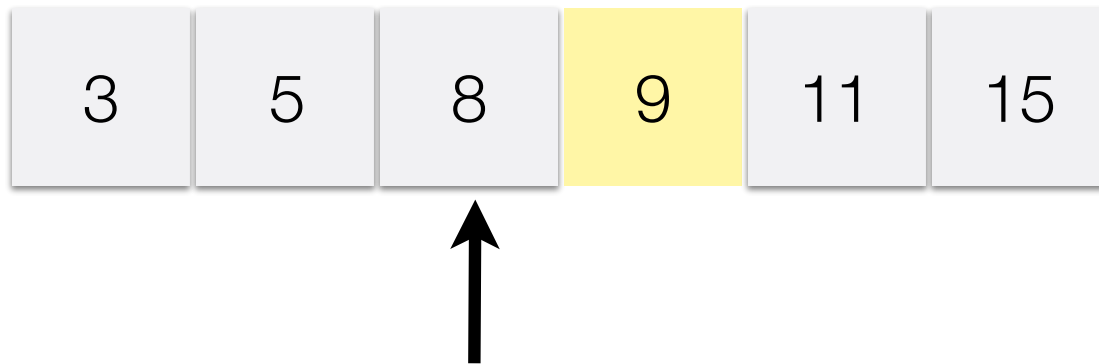


> 9 ?

Insertion Sort

Current

9



Insertion Sort



Insertion Sort

```
public static void insertionSort(List<Integer> list) {  
  
    for (int i = 1; i < list.size(); i++) {  
        // Get current element  
        int cur = list.get(i);  
        int j;  
        // Starting from left neighbour, shift all elements  
        // greater than cur to the right  
        for (j = i - 1; j >= 0 && cur < list.get(j); j--) {  
            list.set(j + 1, list.get(j));  
        }  
        // Now, j points to the first element smaller or  
        // equal to cur, so put cur in slot j+1  
        list.set(j + 1, cur);  
    }  
}
```


Insertion Sort: Analysis

- For a list of length n , main for loop has n steps
- In the worst case, the inner loop shifts all elements left of the current one place to the right

$$1 + 2 + 3 + \dots + (n - 1) + n = \frac{n \cdot (n + 1)}{2}$$

which grows proportionally to n^2 for large n

- Insertion Sort is in $O(n^2)$, or *quadratic*

For very small (or mostly already sorted) inputs, insertion sort is the fastest algorithm